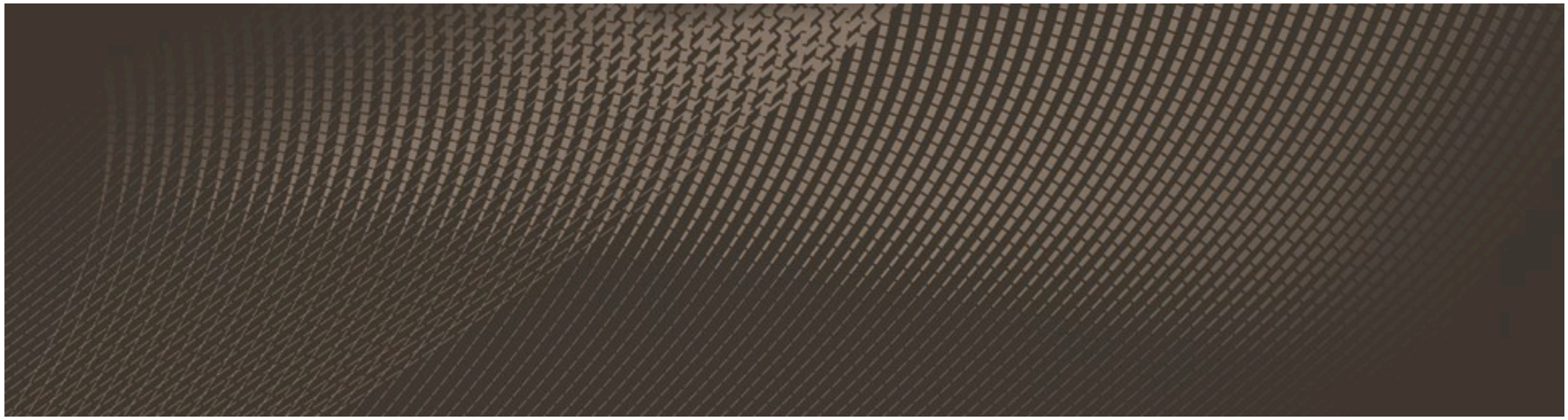


Virtualize and automate your development environment for fun and profit

Andreas Heim - May 23 2011 – Roots Conference, Bergen

BEKK

Vagrant

Virtualize and automate your development environment for fun and profit!

Andreas Heim - BEKK Consulting
@heim

What defines a good development environment?

What defines a good development environment?

- Identical to production

What defines a good development environment?

- Identical to production
- Identical for all developers

What defines a good development environment?

- Identical to production
- Identical for all developers
- Independent of other systems

What defines a good development environment?

- Identical to production
- Identical for all developers
- Independent of other systems
- You should be able to work on a plane

Not this plane

Not this plane

(it has wi-fi)

What defines a good development environment II

What defines a good development environment II

- Lucid

What defines a good development environment II

- Lucid
- Well documented

What defines a good development environment II

- Lucid
- Well documented
- Fast on-boarding of new team members

What defines a good development environment II

- Lucid
- Well documented
- Fast on-boarding of new team members
- Versioned

challenges

team members might develop on two different operating systems

challenges

team members might develop on two different operating systems

and deploy to a third

challenges

developers might have different versions of dependencies

challenges

developers might have different versions of dependencies

database

challenges

developers might have different versions of dependencies

database

build tools

challenges

developers might have different versions of dependencies

database

build tools

app server

challenges

developers might not have taken the time to install

challenges

developers might not have taken the time to install

enterprise database

challenges

developers might not have taken the time to install

enterprise database

enterprise app server

challenges

developers might not have taken the time to install

enterprise database

enterprise app server

because normally “enterprise” means slow
and complex

contamination

if you got multiple applications accessing the same database or central service...

contamination

if you got multiple applications accessing the same database or central service...

how quick can you reset your development environment?

contamination

if you got multiple applications accessing the same database or central service...

how quick can you reset your development environment?

can one application contaminate the data of the other?

automate and virtualize

distribute

virtual machine that contains all of the projects dependencies

distribute

virtual machine that contains all of the projects dependencies

app server, database, stubbed external services

so, what are we actually talking about?

developer 1

OS: snow leopard

some db v. 1.02

app server. v. 9.2.1

apple java 1.6

developer 2

OS: Windows 7

some db v 1.03

app server. v. 9.3.1

sun java 1.6

production machine

OS: Ubuntu

“enterprise db”

“scalable app server”

openjdk 1.5

script the installation of this machine

virtual linux macine

enterprise db
enterprise app server
correct java version

virtualized environment inside your machine

virtualized environment inside your machine

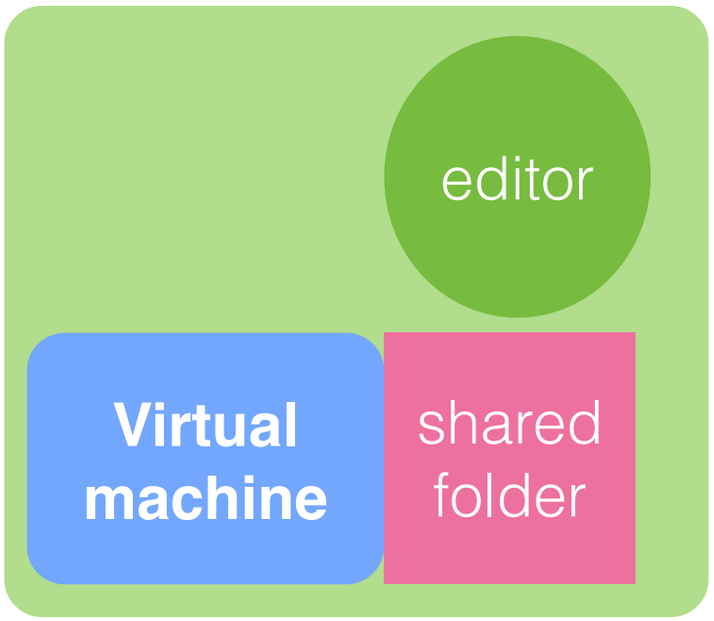
develop using your existing tools

virtualized environment inside your machine

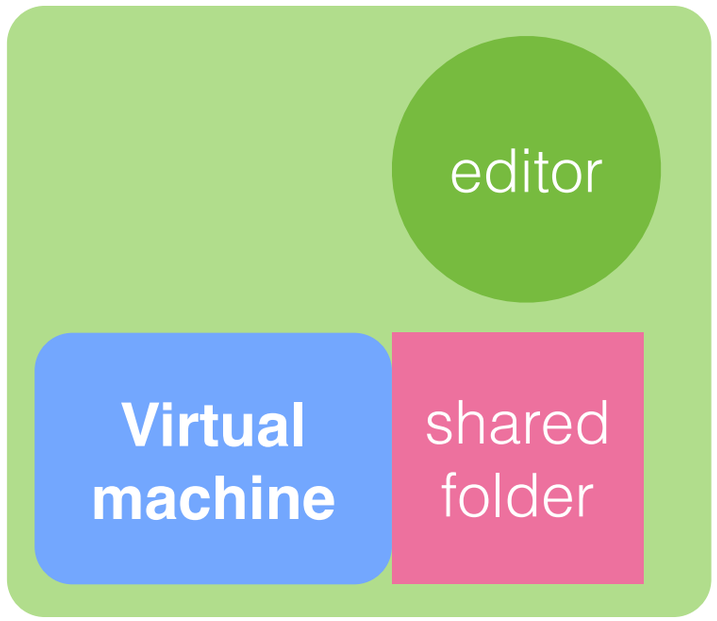
develop using your existing tools

run your code on the virtual machine

developer 1



developer 2



script the setup of this machine

production machine

OS: Ubuntu

vagrant

vagrantup.com

ruby application
(ruby power)
built on top of

ruby application
(ruby power)
built on top of

- sun virtualbox

ruby application
(ruby power)
built on top of

- sun virtualbox
- chef

ruby application
(ruby power)
built on top of

- sun virtualbox
- chef
- puppet

vagrant lets you

vagrant lets you

automate creation
and provisioning
of virtual machines

replicate

replicate

and

rebuild

replicate

and

rebuild

instantly!

built on top of rake

easy to extend

create your virtual machine with

$ vagrant up

create your virtual machine with

$ vagrant up

shut it down with

$ vagrant halt

create your virtual machine with
$ vagrant up

shut it down with
$ vagrant halt

destroy it with
$ vagrant destroy

and rebuild it instantly with

$ vagrant up

easy configuration

```
Vagrant::Config.run do |config|
  config.vm.box = "base"
  config.vm.box_url = "http://files.vagrantup.com/lucid32.box"
end
```

lots of boxes on http://vagrantbox.es

port forwarding

```
Vagrant::Config.run do |config|
  config.vm.forward_port "http", 80, 8080
end
```

forwards port 8080 on the host os to port 80 on the vm

multiple vms

```
Vagrant::Config.run do |config|
  config.vm.define :web do |web_config|
    web_config.vm.box = "web"
    web_config.vm.forward_port("http", 80, 8080)
  end

  config.vm.define :db do |db_config|
    db_config.vm.box = "db"
    db_config.vm.forward_port("db", 3306, 3306)
  end
end
```

provisioning

provisioning

the act of preparing and equipping a VM to run your application

provisioning

the act of preparing and equipping a VM to run your application

in practice, installing software and configuring the machine

alternatives

chef

puppet

alternatives

chef

pure ruby, flexible, big ecosystem

puppet

alternatives

chef

pure ruby, flexible, big ecosystem

complicated, lots of files

puppet

alternatives

chef — pure ruby, flexible, big ecosystem

complicated, lots of files

puppet — supported by google, flexible

alternatives

chef

pure ruby, flexible, big ecosystem

complicated, lots of files

puppet

supported by google, flexible

hard to get started

or use sprinkle, for simplicity

or use sprinkle, for simplicity

sprinkle is a really nice ruby dsl

github.com/crafterm/sprinkle

or use sprinkle, for simplicity

sprinkle is a really nice ruby dsl

github.com/crafterm/sprinkle

```
package :git, :provides => :scm do
  description 'Git Distributed Version Control'
  apt "git-core"
   verify do
    has_executable "git"
  end
end
```

find this code on http://github.com/heim/vagrant-java-example

and roll your own provisioner

```
class SprinkleProvisioner < Vagrant::Provisioners::Base
  def prepare

  end

  def provision!
    vm.ssh.execute do |ssh|
     ssh.exec!('gem list | grep "i18n (0.5.0)" ;if [ $? == "1" ]; then sudo gem install i18n --version "0.5.0"; fi;')
     ssh.exec!('gem list | grep "sprinkle (0.3.3)" ;if [ $? == "1" ]; then sudo gem install sprinkle --version "0.3.3"; fi;')
     ssh.exec!("sudo sprinkle -v -c -s /vagrant/sprinkle/install.rb")
    end
  end
end
```

find this code on http://github.com/heim/vagrant-java-example

config your provisioner for use with vagrant

config your provisioner for use with vagrant

```
Vagrant::Config.run do |config|
  config.vm.provision SprinkleProvisioner
end
```

config your provisioner for use with vagrant

```
Vagrant::Config.run do |config|
  config.vm.provision SprinkleProvisioner
end
```

```
Vagrant::Config.run do |config|
  config.vm.provision :chef_solo
end
```

config your provisioner for use with vagrant

```
Vagrant::Config.run do |config|
  config.vm.provision SprinkleProvisioner
end
```

```
Vagrant::Config.run do |config|
  config.vm.provision :chef_solo
end
```

```
Vagrant::Config.run do |config|
  config.vm.provision :shell, :path => "test.sh"
end
```

but, why?

run code on production like environment

run code on production like environment

instant feedback

run code on production like environment

instant feedback

catches bugs early

run code on production like environment

instant feedback

catches bugs early

even before they hit scm

or if your setup is complicated

or if your setup is complicated

do it quick

or if your setup is complicated

do it quick

reset data quickly

or if your setup is complicated

do it quick

reset data quickly

everything in one package

or if your setup is complicated

do it quick

reset data quickly

everything in one package

and you're probably better off using sprinkle

for teams

for teams

identical environments

for teams

identical environments

single responsibility principle

for teams

identical environments

single responsibility principle

fast onboarding of new team members

for teams

identical environments

single responsibility principle

fast onboarding of new team members

increase in maintainability

for operations (or devops)

for operations (or devops)

test your provisioning scripts

for operations (or devops)

test your provisioning scripts

test your deploy scripts

for operations (or devops)

test your provisioning scripts

test your deploy scripts

test your clustering setup with multi-vms

for operations (or devops)

test your provisioning scripts

test your deploy scripts

test your clustering setup with multi-vms

test load balancing and fallback mechanisms

for operations (or devops)

test your provisioning scripts

test your deploy scripts

test your clustering setup with multi-vms

test load balancing and fallback mechanisms

(this can even be part of your CI-run)

use vagrant if you

use vagrant if you

deploy to the cloud

use vagrant if you

deploy to the cloud

already use chef or puppet

use vagrant if you

deploy to the cloud

already use chef or puppet

have a large amount of dependencies, and a complicated environment setup

your development environment is documented

your development environment is documented

because it is declarative

your development environment is documented

because it is declarative

and it is in the source repository

your development environment is documented

because it is declarative

and it is in the source repository

it is easy to replicate

your development environment is documented

because it is declarative

and it is in the source repository

it is easy to replicate

because its automated

your development environment is documented

because it is declarative

and it is in the source repository

it is easy to replicate

because its automated

it is independent of other systems

your development environment is documented

because it is declarative

and it is in the source repository

it is easy to replicate

because its automated

it is independent of other systems

because its virtualized

thanks!

spkr8.com/heim

@heim

resources:

http://vagrantup.com
https://github.com/heim/vagrant-java-example
https://github.com/crafterm/sprinkle
http://vagrantbox.es

ANDREAS HEIM
CONSULTANT
+47 959 39 833
andreas.heim@bekk.no

BEKK CONSULTING AS
SKUR 39, VIPPETANGEN. P.O. BOX 134 SENTRUM, 0102 OSLO, NORWAY. WWW.BEKK.NO